Ambient-Oriented Programming in AmbientTalk

Stijn Mostinckx* Tom Van Cutsem†
Jessie Dedecker† Wolfgang De Meuter Theo D'Hondt
Programming Technology Laboratory
Department of Computer Science
Vrije Universiteit Brussel, Belgium
smostinc | tvcutsem | jededeck | wdmeuter | tjdhondt@vub.ac.be

ABSTRACT

A new field in distributed computing, called Ambient Intelligence, has emerged as a consequence of the increasing availability of wireless devices and the mobile networks they induce. Developing software for such mobile networks is extremely hard in conventional programming languages because of new distribution issues related to volatile network connections, dynamic network topologies and partial failures.

Categories and Subject Descriptors

D.3.2 [**Programming Languages**]: Language Classifications—*distributed languages*; D.3.3 [**Programming Languages**]: Language Constructs and Features—*Frameworks*

General Terms

Design, Languages

Keywords

ambient intelligence, mobile and ubiquitous computing, actors, language kernel

1. INTRODUCTION

Software development for mobile devices (such as smart phones and PDA's) is given a new impetus with the advent of mobile networks. Mobile networks surround a mobile device equipped with wireless technology (such as Bluetooth or WiFi) and are demarcated dynamically as users move about. Mobile networks turn the applications running on mobile devices from mere isolated programs into smart applications that can cooperate with their environment. As such, mobile networks take us one step closer to the world of ubiquitous computing envisioned by Weiser[8]; a world where (wireless and pervasive) technology is gracefully integrated into the everyday lives of its users. Recently, this vision has been termed Ambient Intelligence (AmI for short) by the European Council's IST Advisory Group.

Mobile networks bring about a number of new distribution issues not found in classical distributed applications [6]. We establish a new paradigm for distributed programming languages that deal with these issues. After describing the properties of such distributed programming languages in general, we illustrate our own programming language called AmbientTalk and hint at how it tackles the new issues.

The Problem: language abstractions for AmI

Mobile networks that surround a device have several properties that distinguish them from other types of networks. First, each device in the network is an autonomous entity with its own processing power. Hence, the natural concurrency of the distributed applications running on top of these devices must be controlled. Second, connections are volatile due to the limited communication range of the wireless technology. Third, networks of mobile devices can form in an ad hoc fashion, which means that the network structure is dynamic (it may change at run-time) and open (devices may join or leave the network unheraldedly). Communication partners often have to be found in an ad hoc fashion, i.e. without any predefined infrastructure.

Dealing with these low-level network concerns at the application-level clutters the code significantly and puts an additional burden on software developers. Although low-level system software and networking libraries (such as JXTA [3] and M2MI [5]) tend to provide uniform interfaces to wireless network technologies, they do not alleviate the task of developing application software for mobile networks. One of the main reasons for this is that contemporary programming languages lack abstractions that avoid the cluttering of application and network-related code. Our goal is to factor out useful abstractions or patterns for programming software deployed in these types of networks and to integrate them into a programming language as language constructs.

The Paradigm: ambient-oriented programming

We feel a new kind of programming paradigm is required to deal with the new issues induced by distribution concerns in mobile computing. That is why we describe a new Ambient-Oriented Programming paradigm [2] (AmOP for

*Author funded by a doctoral scholarship of the "Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen)"
†Research Assistant of the Fund for Scientific Research Flanders, Belgium (F.W.O.)

OOPSLA'05, October 16–20, 2005, San Diego, California, USA.
ACM 1-59593-193-7/05/0010.

short) that consists of programming languages that explicitly incorporate potential network failures in the very heart of their computational model.

For a language to be an ambient-oriented programming language, it should:

offer non-blocking communication primitives. Concretely, this means that messages between objects should be sent *asynchronously*, and that no blocking `receive` operation is available. This constraint ensures that objects will not block waiting for their communication partner for large periods of time, which may be the case due to the volatile, high latency wireless network connections.

be able to reify its communication state. Unreliable communication is a chief concern in ad hoc mobile networks. It is therefore important for a language to enable objects to track the state of outgoing or incoming messages: they should be able to act upon the delivery or arrival of messages, to be informed about the failure of a communication partner, to roll back their state in the case of partial failures, etc. When objects are able to act upon these events, the language has sufficiently reified the object's communication traces.

be able to reify its environment. An AmOP language needs to offer primitives to both *publish* services or objects to the external environment and to *discover* them. This service discovery mechanism is preferably peer-to-peer: it should rely on as little infrastructure as possible. If objects can "sense" other objects on remote devices in their direct environment, an application can effectively reify the hardware environment that surrounds it.

We have embedded these features in our own distributed programming language called AmbientTalk.

The Language: AmbientTalk

AmbientTalk [1] is a first scion of the AmOP programming language family described above. The power of AmbientTalk lies in its simplicity and expressiveness. It offers the programmer features to deal with the conceptual properties of wireless networks without having to deal with their technological characteristics. The basis of AmbientTalk consist of:

- A concurrent object-oriented model combining the power of prototype-based programming as e.g. exemplified by Self [7] and active objects based on the actor model of computation [4].
- A system of first-class message queues (mailboxes) that store an active object's incoming and outgoing messages and which allow the acquisition of services from and provision of services to the environment.
- Reflective properties such as a MOP and a language extension facility which allow one to experiment with new, reflectively implemented language constructs.

An AmbientTalk applications consists of plain, passive, objects and active objects. The active objects are globally addressible and communicate with one another asynchronously. They may provide services to other, remote, active objects and they may require services from the environment. AmbientTalk is an ambient-oriented language: it offers asynchronous message passing and non-blocking receive operations and its first-class mailboxes offer access to incoming and outgoing messages and to other objects (services) in the environment. Although the core set of language primitives is very limited, a lot of more high-level language constructs have been developed reflectively, including future-type message passing, conditional synchronization primitives and weak replication protocols.

More Information

AmbientTalk is presented in a demonstration at OOPSLA05. A technical report describing the language in more detail is available [1]. The AmOP paradigm is presented in more detail in the OOPSLA05 Onward! presentation track [2].

2. REFERENCES

[1] Dedecker, J., Van Cutsem, T., Mostinckx, S., De Meuter, W., and D'Hondt, T. Ambienttalk: A small reflective kernel for programming mobile network applications. Tech. Rep. VUB-PROG-TR-05-06, Programming Technology Laboratory, Department of Informatica, Vrije Universiteit Brussel, 2005.

[2] Dedecker, J., Van Cutsem, T., Mostinckx, S., D'Hondt, T., and De Meuter, W. Ambient-oriented programming. In *OOPSLA '05: Companion of the 20th annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications. San Diego, U.S.A. ACM Press.* (2005), ACM Press.

[3] Gong, L. JXTA for J2ME extending the reach of wireless with JXTA technology. Tech. rep., SUN Microsystems, http://www.jxta.org/project/www/docs/JXTA4J2ME.pdf, 2002.

[4] Hewitt, C. E. Viewing control structures as pattern of passing messages. *Artificial Intelligence: An International Journal 8*, 3 (June 1977), 323–364.

[5] Kaminsky, A., and Bischof, H.-P. Many-to-many invocation: A new object oriented paradigm for ad hoc collaborative systems. *17th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA 2002)* (2002).

[6] Mascolo, C., Capra, L., and Emmerich, W. Mobile computing middleware. In *Advanced lectures on networking*, vol. 2497. Springer-Verlag New York, Inc., 2002, pp. 20–58.

[7] Ungar, D., and Smith, R. B. Self: The power of simplicity. In *Conference proceedings on Object-oriented programming systems, languages and applications* (1987), ACM Press, pp. 227–242.

[8] Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (1991), 66–75.